International Journal for Computational Biology (IJCB)
Vol.2, No.1, 2013, Page: 01-11
ISSN: 2278-8115

A Novel DNA Sequence Compression Method Based on Chaos Game Representation

Arun K. S*, Achuthsankar S. Nair, Oommen V. Oommen
Department of Computational Biology and Bioinformatics, University of Kerala, India

Article Info

Article history:

Received Feb 6th, 2013
Revised Apr 24th, 2013
Accepted Jun 9th, 2013

Keyword:

Chaos Game Representation
Sequence Compression
Information-Content
Information-Entropy

ABSTRACT

Unique signature images derived out of Chaos Game Representation of bio-sequences is an area of research that has been confined to pattern recognition applications. In this paper we pose and answer an interesting question – can we reproduce a bio-sequence in a lossless way given the co-ordinates of the final point in its CGR image? We show that it is possible in principle, but would need enormous resolution for representation of coordinates, roughly corresponding to the information content of direct binary coding of the sequence. We go on to show that we can code nucleotide codon triplets using this method in which 16 codons can be coded using 4 bits, the remaining 48 using 6 bits. Theoretically up to 11% compression is possible with this method. However, algorithm overheads reduce this to very nominal compression percentage of less than 4% for human genome and 9% for bacterial genome. We report the results on a subset of standard test sequences and also an independent wider data set.

Corresponding Author:

Arun K. S,
Department of Computational Biology and Bioinformatics, North Campus - Kariavattom, University of Kerala, Thiruvananthapuram, Keralam, India – 695581
Email: arunksreedhar12@gmail.com

How to Cite:

Arun K. S *et. al.* A Novel DNA Sequence Compression Method Based on Chaos Game Representation. 2013; Volume 2 (Issue 1): Page 01-11.

1. INTRODUCTION

During 1970s, a new field of physics was developed known as chaotic dynamical systems or simply chaos [1]. This field is closely associated with fractals. Fractal geometry, in contrast with Euclidean geometry, deals with objects that possess fractional dimensions. Fractal geometry considers itself the geometry of the real (rather than the ideal) and consequently treats the objects in nature as possessing fractal dimensions. Among interesting properties of the fractals are their unvarying complexities at varying scales. The Chaos Game is an algorithm, which is an offshoot of research in the above area. It allows one to produce unique images of fractal nature, known as Chaos Game Representation images (CGR images) from symbolic sequences, which can serve as signature images of the sequences. It was originally described by MichealBarnsley in 1988 [2]. Chaos Game is an algorithm whose input is a sequence of letters (finite alphabets) and output is an image (see Fig. 1). Biological sequences like DNA, RNA and amino acid sequences can be represented by sequence of finite alphabets, are amenable to conversion to CGR images. The use of CGRs as useful signature images of bio-sequences such as DNA has been investigated since early 1990s. CGR of genome sequences was first proposed by H. Joel Jeffrey [2]. Later other bio-sequences were also explored. We will now briefly introduce the idea of deriving a CGR image of a DNA sequence. To derive a Chaos Game Representation of a genome, a square is first drawn to any desired scale and corners marked A, T, G and C. The choice of the corners is not based on any particular criteria, and indeed can be assigned in any other way. Points are marked within the square corresponding to the nucleotides in the sequence.

Nucleotide A, T, G and C have assigned positions (0, 0), (1, 0), (1, 1) and (0, 1) respectively (see Fig. 2). The centre P0 of the CGR square is (0.5, 0.5). Now we define a procedure for representing any arbitrary

nucleotide sequence as points inside the square. For plotting a given sequence, we start from the centre of the square. The first point is plotted halfway between the centre of the square, and the corner corresponding to the first nucleotide of the sequence, and successive points are plotted halfway between the previous point, and the corner corresponding to the base of each successive nucleotide. The steps for plotting a given sequence are concluded below.

1. Select the first nucleotide from the given sequence.
2. Calculate the midpoint between the centre and the corner corresponding to the first nucleotide (xN, yN). Let the midpoint be (xi, yi). Let (xc, yc) be the co-ordinates of the midpoint of the square.
Then **xi = (xc+ xN)/2 and yi= (yc+ yN)/2**
3. Do the following steps until all the nucleotides are processed: Read the next nucleotide in the sequence. Calculate the midpoint between the current point (xi, yi) and the corner corresponding to the newly read nucleotide: **xi+1 = (xi+ xN)/2 and yi+1 = (yi+ yN)/2.**

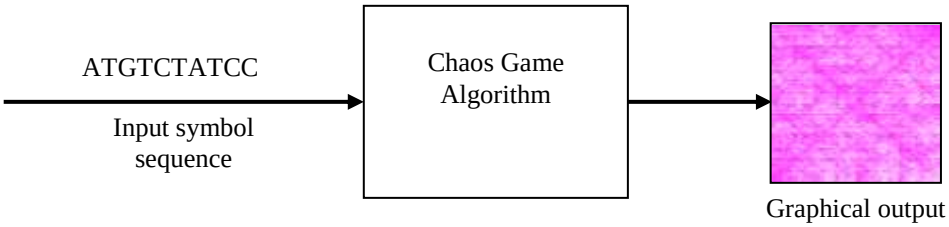

Figure 1. Chaos Game Representation process.

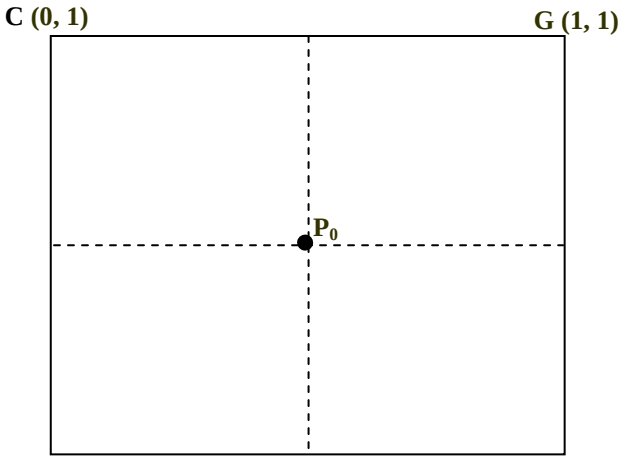

Figure 2. CGR square with each nucleotide assigned to corners

Now using the above procedure let us plot a DNA sequence TACAGA into the CGR square. Points are marked within the square corresponding to the bases in the sequence, as follows:

1. Plot the first point P1, halfway between the center of the square P0, and the T corner.
2. The next point P2 is plotted halfway between P1 and the A corner.
3. The next point P3 is plotted halfway between P2 and the C corner.
4. The next point P4 is plotted halfway between P3 and the A corner.
5. The next point P5 is plotted halfway between P4 and the G corner.
6. The next point P6 is plotted halfway between P5 and the A corner.

Figures 3 depict the process graphically.

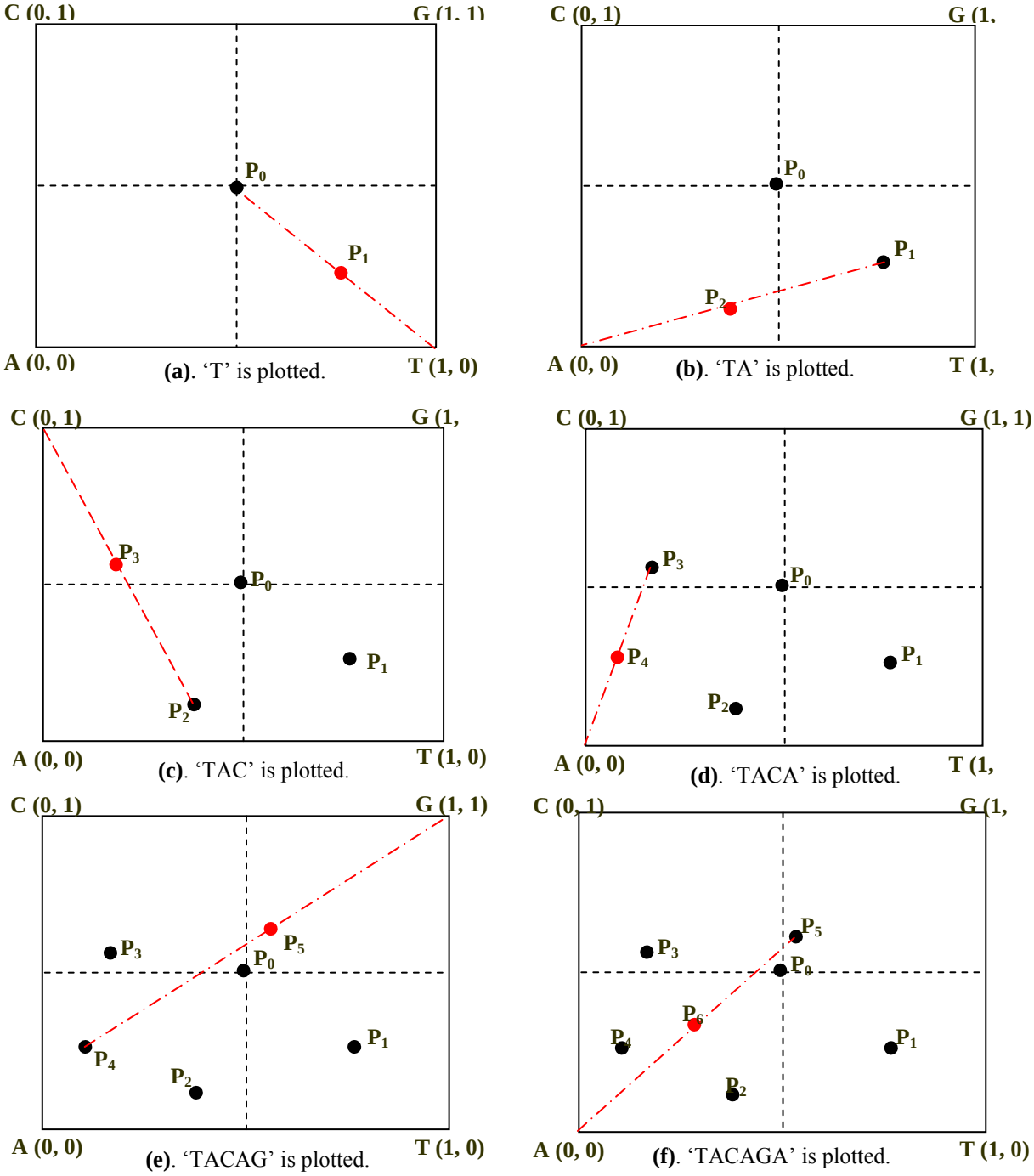

Figure 3. Plot of **CGR** points for the system "TACAGA"

Now, let us see how a real CGR would look like. Fig. 4 shows CGR of whole genome of *Thermosinuscarboxydivorans Nor1* (NCBI accession code: NZ_AAWL00000000.1), plotted using a tool C-GRex developed by the authors [3]. A CGR has many interesting properties. Every bio-sequence has a unique CGR. In fact every symbol in a sequence will have a corresponding unique point in the CGR, even though the reverse need not be the case. Every point on the CGR is a representation of all the symbols in the sequence up to that point. For instance, in the CGR of the sequence ATTTGGCCATCG, the fifth point represents the sequence ATTTG. Each sub-square in a CGR has a special significance. If we divide the CGR into four quadrants, then the top right corner will contain points representing sub-sequences that end with G, as a midpoint between any other point in the square and the G-corner has to fall in this quadrant. Hence if we count

the points in this quadrant, it will be equal to the count of the base G in the sequence. If we divide this quadrant into another 4 squares, in the clockwise order, they would represent subsequences that end in GG, TG, AG and CG, making it possible to derive the 2-mer counts by counting the points in these sub-squares. In general, by dividing the CGR square into sub-squares of side 2–n, we can find the number of different n-mers present in the sequence.

2. RESEARCH METHOD

2.1 Reversing the Chaos Game Representations

Many works can be seen on investigation into use of CGRs as unique signature images for genomes and also other bio-sequences [4] [5] [6] [7] [8]. This paper however addresses a slightly different question. The CGR images are derived out of DNA sequences. Can the sequence be reconstructed given its CGR? More specifically, can we reconstruct the sequence up to a particular point, given the coordinates of the final point in the CGR image? Our interest in the question arises from the recognition of the possibility that if the answer to the question we raised is yes, then we have a unique and arguably efficient way of compressing DNA sequences. If the answer is an unqualified yes, then every DNA sequence, irrespective of its length can be represented by two numbers, the coordinates of the last nucleotide of the DNA sequence when it is plotted in the CGR image. We will demonstrate the basic idea with a toy example. Let us consider a DNA sequence TACAGAAACG, of length 10 bases, and we show below in Table 1 the x and y coordinates of the pixels plotted in the CGR image of this sequence.

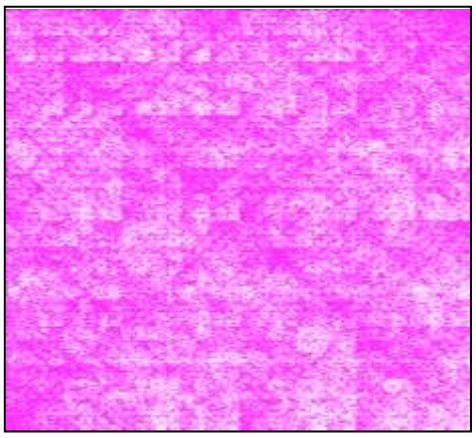

Figure 4.CGR of *Thermosinuscarboxydivorans* Nor 1 whole genome.

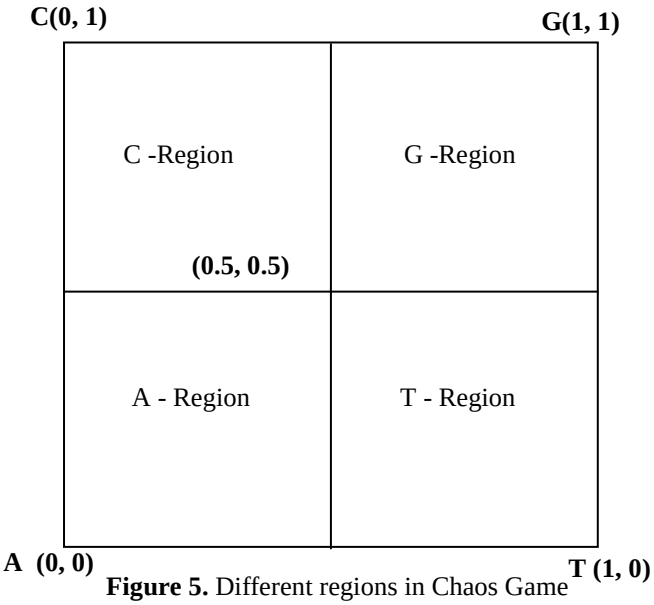

Figure 5. Different regions in Chaos Game Representation

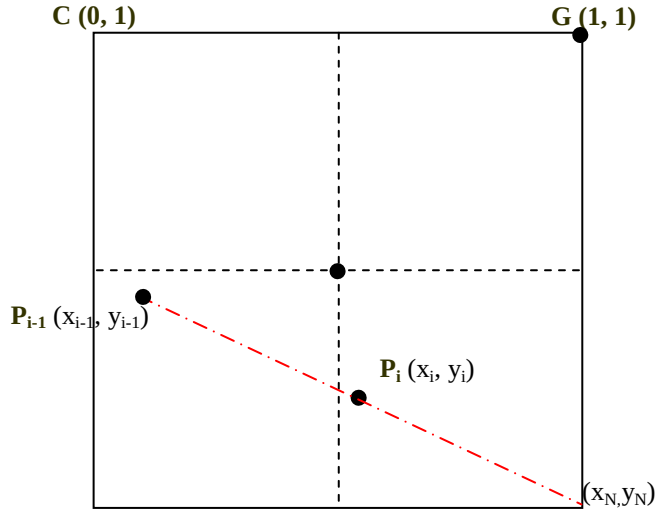

Figure 6. Tracing back from CGR to square

Now we apply our general question that we raised above in this specific example. Given the coordinates of the last nucleotide T (0.5170898, 0.7700195), can we trace back the whole sequence? A cursory look at the CGR image plotting procedure would tell us that it is a rather trivial matter to do this trace back. One of the main properties of CGR square is that, all DNA sequences which end with a particular nucleotide will be plotted within the corresponding region. For example, all DNA sequences which terminate in A will fall in the A-Region. Fig 5 shows all regions in a CGR square.

Thus, if we know the region in which a given point falls, then we can find the corresponding nucleotide. Further, if we know the 'i'th coordinates of a sequence then we can calculate the coordinates immediately preceding it. i.e. (i-1)th point. This can be done exploiting the fact that (i-1)th point is the midpoint of ith point and the corner specified by that nucleotide.

Table 1. CGR Coordinates of the 10-base sequence 'TACAGAAACG'

Sequence	X Coordinate	Y Coordinate
T	0.75	0.25
TA	0.375	0.125
TAC	0.1875	0.5625
TACA	0.09375	0.28125
TACAG	0.546875	0.640625
TACAGA	0.2734375	0.3203125
TACAGAA	0.1367187	0.1601562
TACAGAAA	0.0683593	0.0800781
TACAGAAAC	0.0341796	0.540039
TACAGAAACG	0.5170898	0.7700195

Consider a nucleotide sequence S. Let the coordinates corresponding to the 'i'th nucleotide Ni be (xi, yi). Let (xN,yN) be the coordinate of the CGR corner in which the 'i'th nucleotide falls. Since (xi, yi) is the midpoint of (xN,yN) and (xi-1, yi-1), we have xi = (xi-1 + xN) /2 and yi = (yi-1 + yN) /2 . It is trivial to note that: xi-1 = 2 xi- xN and yi-1 = 2 yi - yN. Fig. 6shows the points (xi, yi), (xN,yN) and (xi-1, yi-1). Since (xi, yi)

is in T-Region, the nucleotide at position i is T. Similarly (xi-1, yi-1) is in A-Region, so the nucleotide at position 'i-1' is A. Thus we can trace back the whole sequence.

We give below the algorithm for trace-back of nucleotides, from final CGR coordinate. Let the corners of the square be A at (0, 0), T at (1, 0), G at (1, 1) and C at (0, 1). Let S be an array that stores the generated sequence. Let (x1, y1) be the current coordinate point and let (xN,yN) be the coordinate of the relevant CGR corner.

```
start
initialise (x1, y1) to the last coordinate point of the CGR of given sequence
i=1;
while(x1 !=0 && y1 !=0)
{
        r = findRegion(x1, y1);
        if r == 1 { push 'A' to S; xN = 0, yN = 0; }
        else if r == 2 { push 'T' to S; xN = 1, yN = 0; }
        else if r == 3 { push 'G' to S; xN = 1, yN = 1; }
        else if r == 4 { push 'C' to S; xN = 0, yN = 1; }
x1 = 2 * x1 - xN;
y1 = 2 * y1 - yN;
i = i + 1;
}
n=i;
reverse (S);
stop
```

The procedure findRegion accepts two parameters corresponding to a coordinate position and returns a number representing the region (1, 2, 3, and 4 for A, G, T and C regions respectively) in which the specified coordinate point lies. The procedure findRegion is called by the above algorithm in each iteration to predict the correct nucleotide.

```
intfindRegion(int x, int y)
{
if (x<0 & y<0) { return 1; }
if (x>0 & y<0) { return 2; }
if (x>0 & y>0) { return 3; }
if (x<0 & y>0) { return 4; }
}
```

The application of the above algorithms to the toy sequence 'TACAGAAACG' is shown in Table 2. If we read from bottom to top we will get the DNA sequence corresponding to the given point. Thus we can regenerate the complete sequence from the last coordinate.

Now our demonstration on the trivial example shows that the DNA sequence TACAGAAACG can be represented in a lossless manner by the (x, y) coordinates of the last nucleotide G. We have definitely produced a unique way of representing the sequence. Can we compress the DNA sequence based on this? We observe (see Table 1) that resolution of the CGR coordinates increases as the length of the sequence increases. This length has to be weighed against the direct binary representation of nucleotides to explore possibility of compression. Direct binary representation of 4 nucleotides requires log2 (4) = 2 bits/nucleotide. If the CGR coordinates of a sequence of 10 bases is represented using 4-byte floats, then this representation costs 64 bits whereas direct binary coding requires only 20 bits. We find that in the 10 base sequences, the coordinates of the final point in CGR has prominent decimal places upto7 positions. Thus our attempt to compress the sequence is immediately faced with a newer limit -- the resolution of the CGR coordinate. We propose an approach to compress DNA sequence by facing this limitation. Before we present our method to achieve compression of DNA sequences, we briefly review the existing methods for the same.

One of the first DNA sequence compression algorithm was reported in 1993-94 by Grumbach and Tahi. Two lossless compression algorithms for DNA sequences proposed by them are Biocompress andBiocompress-2 [9]. These two algorithms are based on the Lempel-Ziv data compression method. Biocompress-2 first detects exact repeats and complimentary palindromes in the sequence. These detected repeats and palindromes in the target sequence are then encoded. E. Rivals reports another compression algorithm named Cfact[10], whichsearches the longest exact matching repeat using suffix tree data structure in an entire sequence. The GenCompress algorithm, introduced in 1999 by Xin Chen and Sam Kwang [11] yields

significantly better compression ratio than the previous algorithms. The idea is to use approximate instead of exact repetitions. CTW+LZ [13] is another DNA compression algorithm based on context free weighting method. It is a combination of GenCompress and CTW. Long exact/approximate repeats are encoded by LZ77 type algorithm and shorter repeats are encoded by CTW method. This algorithm provides very good compression ratio. But the execution time required is too high for long sequences. DNACompress employs the Lempel Ziv compression scheme [12].The method consists of two phases. In the first phase it finds all approximate repeats and complimentary palindromes using specific software PatternHunter. During the second phase the approximate repeats and non-repeats are encoded. DNAC is yet another method by Chang C. H. [14]. DNAC is a DNA compression tool having four phases. During the first phase it builds a suffix tree to locate exact repeats. In the second phase all exact repeats are extended to approximate repeats by dynamic programming. During the third phase it extracts the optimal non-overlapping repeats from overlapping ones. In the last phase the algorithm encodes all repeats. DNASequitur[16] is a grammar-based DNA compression algorithm which infers a context free grammar to represent the input data. Even if the algorithm is elegant, its performance is not on par with other methods. DNAPack [15] uses Hamming distance for the repeats & complimentary palindromes and CTW for non-repeat groups. DNAPack uses a dynamic programming approach for choosing the repeats instead of greedy method. GergelyKorodiet. al. [17] report a new method for compressing DNA sequences using normalized maximum likely hood.

Table 2. Recursive derivation of sequence from CGR coordinates

Current X Coordinate x_i	Current Y Coordinate y_i	Nucleotide	Corner Vertex (x_N,y_N)	Previous X Coordinate $x_{i-1} = 2x_i - x_N$	Previous Y Coordinate $y_{i-1} = 2y_i - y_N$
0.5170898	0.7700195	G	(1, 1)	0.0341796	0.540039
0.0341796	0.540039	C	(0, 1)	0.0683592	0.080078
0.0683592	0.080078	A	(0, 0)	0.1367184	0.160156
0.1367184	0.160156	A	(0, 0)	0.2734368	0.320312
0.2734368	0.320312	A	(0, 0)	0.5468736	0.640624
0.5468736	0.640624	G	(1, 1)	0.0937472	0.281248
0.0937472	0.281248	A	(0, 0)	0.1874944	0.562496
0.1874944	0.562496	C	(0, 1)	0.3749888	0.124992
0.3749888	0.124992	A	(0, 0)	0.7499776	0.249984
0.7499776	0.249984	T	(1, 0)	0.4999552	0.499968

All the methods described above report their results on almost entirely different set of data and hence it is impossible to compare the results directly. Most of the tools are not any more available for testing online or otherwise. Hence we are unable to report a comprehensive comparison. For the sake of comparison with our work, we have chosen to separate out the common sequences on which test results have been reported in all the paper reviewed above. It is not clear whether the dataset on which results are reported in the above papers have been hand-picked or randomly chosen. Sequences with repeats and inverted repeats achieve good compression ratios, but test results on random data are required to conclude further about the existing methods.

2.2 Compressing DNA sequence using CGR coordinate reversing

Based on our experimentation, we have found that it is ideal to consider the sequence as a string of triplet bases (codons) and reduce the problem to that of compressing codons only. With four bases, there are 64 codons and assuming equal probability for all codons, the number of bits required will be $2 \times 3 = 6$ bits/ codon. Through actual experimentation, we found that 16 of the 64 codons can be derived by reversing the CGR coordinates, with only 2 bits representation for each co-ordinate. This means that we can represent those using 4 bits/codon. The remaining 48 codons require 6 bits, but the informational space is not fully utilized in these codes as, $\log_2 48 = 5.85$ (in place of 6). This extra space can be utilized for implementing overheads of the compression algorithm proposed below. Table 4 shows the code assigned for 64 codons.

Our compression process works as follows. We first slice the DNA sequence to be compressed into codons. Corresponding binary sequence is generated based on codes assigned. In this binary sequence, some codons are coded by 4 bits and others by 6 bit code. When we encode a codon by 4 bit, we are compressing the codon by 4/6 *100 =33%. The information whether a codon is coded by 4 or 6 bit is either stored in the dual code of the previous codon or using a bit in a separate binary sequence. The algorithm for compression is given below. The sequence length is assumed to be a multiple of 3. (Otherwise, the remaining 1 or 2 nucleotides can be coded using 8 bits and this situation can be trivially handled). The algorithm accepts a DNA sequence as input and produces two binary sequences. First binary sequence is compressed DNA and second one contains some information regarding compressed binary sequence.

Compression Algorithm
DNASeq – String array storing input DNA sequence.
CompDNASeq – Binary array storing compressed DNA.
InfoSeq – Bit array indicating 4-bit/6-bit coding using 0/1.
start
whileDNASeq<> null do
{
read the next codon
if a 4 bit code exists:
append 4 bit code to CompDNASeq; Append '1' to InfoSeq.
else if two six bit codes exists
read the next codon
if it has a 4 bit code
append the second 6 bit code of the previous codon to CompDNASeq.
append the 4 bit code of the current codon to CompDNASeq.
append '0' to InfoSeq.
else
append the 6 bit code of the previous codon to CompDNASeq.
append '0' to InfoSeq.
move the cursor one codon backward.
end
end
}
stop

Now we give below the decompression algorithm. This takes the two binary arrays produced by the compression algorithm and derives the original DNA sequence.

Decompression Algorithm
CompDNASeq – Binary sequence array which stores the compressed DNA.
InfoSeq – Bit array storing whether 4-bit/6-bit codes are used for each codon.
DNASeq – String array storing decompressed DNA sequence is stored.

start
do the following steps until all codons are processed.
{
if the current bit in InfoSeq is 0
take the next six bits from CompDNASeq.
if it is greater than 01 11 11
append the corresponding codon to DNASeq.

take the next four bits from CompDNASeq.
append the corresponding codon to DNASeq.
else
append the corresponding codon to DNASeq.
end.
else
take the next four bits from CompDNASeq.
append the corresponding codon to DNASeq.
end.
}
saveDNASeq file.
stop

Tables and Figures are presented center, as shown below and cited in the manuscript.

3. RESULTS AND ANALYSIS

We implemented the CGR-based DNA sequence compression algorithm and tested it on two sets of data. To enable comparison with existing methods, we have used the common data indicated in Table 4 first. The results appear in the Table 5. Further we also report results in an enlarged and more representative dataset, in Table 6. The maximum compression achieved is nearly 8.8% for bacterial gene 9128976 (lpa_01672). If 4-bit codons are less than approximately 36%, then the sequence is not compressible.

Table 3. CGR codes assigned for codons of Bacterial genome. 6 bit codes are used to code only 48 codons. Hence same codons are assigned dual codes which can be used to store one extra bit, for algorithm implementation overheads.

Sl. No.	Codon	Code	Sl. No.	Codon	Code	Sl. No.	Codon	Code	Sl. No.	Codon	Code
1	AAA	0000	17	GAC	100000 110000	33	TAG	000000	49	GTA	010000
2	ATA	0100	18	GCA	100001 110001	34	TGA	000001	50	CTT	010001
3	AGA	0101	19	CGT	100010 110010	35	AGG	000010	51	AAG	010010
4	ACA	0001	20	AAC	100011 110011	36	TAA	000011	52	TAC	010011
5	TAT	1000	21	CCG	100100 110100	37	CGA	000100	53	CAT	010100
6	TTT	1100	22	ACC	100101 110101	38	CTA	000101	54	TTG	010101
7	TGT	1101	23	ATC	100110 110110	39	CGG	000110	55	TTA	010110
8	TCT	1001	24	GCC	100111 110111	40	TGC	000111	56	ACG	010111
9	GAG	1010	25	GGT	101000 111001	41	CCT	001000	57	TGG	011000
10	GTG	1110	26	ATG	101001 111001	42	TCA	001001	58	CAA	011001
11	GGG	1111	27	GGC	101010 111010	43	CCA	001010	59	GTC	011010
12	GCG	1011	28	CAG	101011 111011	44	GGA	001011	60	AGC	011011
13	CAC	0010	29	ATT	101100 111100	45	TCG	001100	61	GCT	011100
14	CTC	0110	30	GAT	101101 111101	46	TCC	001101	62	TTC	011101
15	CGC	0111	31	GAA	101110 111110	47	AGT	001110	63	AAT	011110
16	CCC	0011	32	CTG	101111 111111	48	ACT	001111	64	GTT	011111

At the outset, we note that the results of the two popular DNA compression tools – Gencompress1 and DNA Compress (which are the ones available on-line for testing) both accept ASCII text files as inputs and show compression on the ASCII text. However, DNA text files (we ignored the headers, they can be trivially managed) need only 2 bits/base as there are only 4 symbols in the string. If we consider this fact, then both the above methods are seen to show very unimpressive performance. Hence comparison with our method is not directly possible. As the test sequences of the previously reported algorithms are not available, no exact comparison and conclusion is possible. Our testing with these tools revealed that they mostly expand the data files, sometimes to even 9bits/base. In the case of our algorithm, we note that the effectiveness of this algorithm is dependent on the occurrence of 4 bit codons in the sequence. It is possible to predict the compressibility of the sequence by computing the probability of 4-bit and 6-bit codons (as in Table 4). The expected bits/codon is $p_4 \times (4+1) + p_{6a} \times 6 + p_{6b} \times (6+1)$, where p_4 = probability of 4-bit codons (+1 is for overhead in InfoSeq), p_{6a} = probability of 6-bit codons (overhead is not required, as dual codes can serve the purpose), and p_{6b} = probability of 6-bit codons (+1 is for overhead in InfoSeq). We achieved best case compression of 1.82 bits/base and worst case of only 2.082 bits/base.

Table 4. Claimed Test results of the algorithm for DNA compression reported in literature since 1991.Only the results on sequences which are commonly reported in all the reviewed papers are included in the table. The sequences included are: chloroplast genomes (GenBankID: X04465, and Z00044), mitochondria genomes (M68929 and X55026), human genes (M86524, J03071 and M26434) and two viral genomes (X17403 and M35027). It is to be noted that the claimed results could not be verified in most cases. In some cases, our verification showed these figures to be in general not achievable.

Sl no	Method	Key Approach	Compression achieved (bits/nucleotide)		
			Best	Worst	Average
1	BioCompress2 [12][16][18]	Lampel-Ziv algorithm	1.31	1.94	1.78
2	Cfact [12][16][18]	Suffix tree	1.49	1.93	1.71
3	GenCompress2 [12][16][18]	Edit distance	1.1	1.92	1.74
4	CTW+LZ [12][16][18]	Context free weighting	1.1	1.92	1.74
5	DNACompress[12][16][18][19]	Lampel-Ziv algorithm	1.03	1.91	1.73
6	DNAPack [12][16][18]	Hamming distance and CTW	1.04	1.90	1.71
7	GeMNL [12][16][18]	Adaptive Markov models	1.01	1.91	1.66

The significance of information entropy is that it tells us the minimum number of bits required to encode the message digitally [20]. This would mean that if we measure the entropy of a message, we know if there is scope for compression of the message. For example, a DNA sequence of four letters, A, G, C and T. A simple coding mechanism is to code each letter with two bits (as there are four signals, $\log_2 4=2$). However, if it is known that probability of symbols A,G,C and T are 0.15,0.35,0.15 and 0.35 respectively then

$H = - \sum p_i \log p_i$

= - (0.15*log(0.15) + (-) 0.35*log(0.35) + 0.15*log(0.15) + 0.35*log(0.35))

= 1.88b.

This would mean that this DNA sequence can be compressed ((2-1.88)/2)=5.9%. Amino acid sequences, which some researcher claim as incompressible [18], can be compressed using our algorithm by simply reverse translating amino acid sequences to corresponding DNA sequences.

4. CONCLUSION

In addition to proposing a unique way of compressing DNA sequences, the work reported in this paper also throws up many new ways of approaching sequence studies. Whether sequences can be compared based on final CGR coordinates is one such problem of exploration. Intuitively, the difference between final coordinates indirectly would indicate the amount of evolutionary divergence. Further studies are required in this direction to confirm if this would be practical and if it would be advantageous over existing scoring scheme.

It is also noted here that the algorithmic implementation has scope for further refinement. At an average, a DNA sequence has about 30% of 4 bit codons which can be compressed by 33%. Thus 11% compression is achievable at an average. Further refinement in algorithm implementation is required to enhance this currently achieved compression results to reach this threshold.

We have demonstrated in this paper that DNA sequences can be compressed by representing the codons in them by using CGR coordinates. The best compression rates achieved are 1.82bits/base against the 2 bit/base standard coding. This compression rates are reasonable, given the limits imposed by Shannon's entropy. In the near future, when personal genome sequencing may become indispensable, storing DNA sequence in handhold devices may be required. The work reported in the paper opens a new way for realizing this application.

ACKNOWLEDGEMENTS

We express our sincere thanks to Dr. Pawan K. Dhar, Hon. Director, Centre for systems and Synthetic Biology, University of Kerala for the helpful discussions. This work has its roots in a project funded by Department of Information Technology, Govt. of India (DIT/R&D/B10/15(23)2008, dated 07/09/2010).

REFERENCES

[1]Kathleen T. Alligood , Tim D. Sauer , James A. Yorke , "Chaos: An Introduction to Dynamical Systems", Springer (1997).
[2] Jeffrey H J Chaos game representation of gene structure. Nucleic Acids Res. 18:2163–2170 (1990).
[3] Achuthsankar S Nair, Vrinda V Nair, K S Arun, Krishna Kant, AlpanaDey Bio-sequence Signatures Using Chaos Game Representation. In: Fulekar M H editor. Bioinformatics: Applications In Life And Environmental Sciences Springer Netherlands 62-76 (2010).
[4] Deschavanne P J, Giron A, Vilain J, Fagot G, Fertil B Genomic signature: characterization and classification of species assessed by chaos game representation of sequences. Mol. Biol. Evol. 16:1391-1399 (1999).
[5] Antonio Neme, Antonio Nido, VíctorMireles, Pedro Miramontes The self-organized chaos game representation for genomic signatures analysis. Learning and Nonlinear Models Revista da SociedadeBrasileira de RedesNeurais (SBRN) 6: 111-120 (2008).
[6] Joseph J, Sasikumar R Chaos game representation for comparison of whole genomes. BMC Bioinformatics 7:243 (2006).
[7] Bai-linHao, H C Lee, Shu-yu Zhang Fractals related to long DNA sequences and complete genomes. Chaos, Solitons& Fractals 11: 825-836 (2000).
[8] Vrinda V Nair, KarthikaVijayan, Deepa P Gopinath, Achuthsankar S Nair ANN Based Classification of Unknown Genome Fragments Using Chaos Game Representation. Proc of the Second International Conference on Machine Learning and Computing IEEE Computer Society Washington DC USA 81-85 (2010).
[9] Grumbach S, Tahi F A new challenge for compression algorithms: Genetic sequences. J. Inform. Process. Manage. 30: 875–886 (1994).
[10] E Rivals, J P Delahaye, M Dauchet, and O Delgrange A guaranteed compression scheme for repetitive DNA sequences. LIFL Lille I Univ Tech. Rep. IT–285 (1995).
[11] Chen X, Kwong S, Li M A compression algorithm for DNA sequences. IEEE Engineering in Medicine and Biology.61–66 (2001).
[12] Chen X, Li M, Ma B, Tromp J DNA Compress: fast and effective DNA sequence compression. Bioinformatics 18:1696–1698 (2002).
[13] Matsumoto T, Sadakane K, Imai H Biological sequence compression algorithms. In Genome Informatics Workshop Universal Academy Press 43–52 (2000).
[14] Chang C H DNAC: A Compression Algorithm for DNA Sequences by Nonoverlapping Approximate Repeats. Master Thesis. National Taiwan University, Graduate Institute of Information, Taipei, Taiwan (2004).
[15] B Behzadi, F L Fessant DNA compression challenge revisited: A dynamic programming, approach. CPM 190–200 (2005).
[16] Neva Cherniavsky, Richard Ladner Grammar-based Compression of DNA Sequences. 2004, UW CSE Technical Report 2007-05-02(2004) .
[17] KorodiG,Tabus I, Rissanen J, Astola J DNA sequence compression - Based on the normalized maximum likelihood model. Inst. of Signal Process. Tampere Univ. of Technol. Signal Processing Magazine IEEE 24: 47 – 53 (2007).
[18] Craig G. Nevill-Manning, Ian H. Witten Protein is Incompressible. DCC '99 Proceedings of the Conference on Data Compression IEEE Computer Society Washington, DC, USA 257-266.
[19] Minh Duc Cao, Trevor I. Dix, Lloyd Allison, Chris Mears (2007) A Simple Statistical Algorithm for Biological Sequence Compression. IEEE Data Compression Conference, Snowbird, UT 42-45 (1999).
[20] AchuthsankarS.Nair, Arun K.S .It's 60 years since"KPB WCY XZ" became more informative than "I LOVE YOU" IEEE Potentials (ISSN: 0278-6648) vol.29, Issue 6, pp.16-19,Nov-Dec (2010).